(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 December 2005 (08.12.2005)

PCT

(10) International Publication Number
WO 2005/117372 A1

(51) International Patent Classification[7]: H04L 12/58, 29/08

(21) International Application Number:
PCT/NO2005/000175

(22) International Filing Date: 27 May 2005 (27.05.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
20042233 28 May 2004 (28.05.2004) NO

(71) Applicant *(for all designated States except US)*: **TELENOR ASA** [NO/NO]; Snarøyveien 30, N-1331 FORNEBU (NO).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **DO, Thanh Van** [NO/NO]; Stjernemyrveien 28, N-0673 OSLO (NO). **MOE, Jon-Finngard** [NO/NO]; Heimvang 1, N-2770 JAREN (NO). **SIVERTSEN, Eivind** [NO/NO]; Furuveien 15, N-1357 BEKKESTUA (NO).

(74) Agent: **OSLO PATENTKONTOR AS**; P.O. Box 7007M, N-0306 OSLO (NO).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

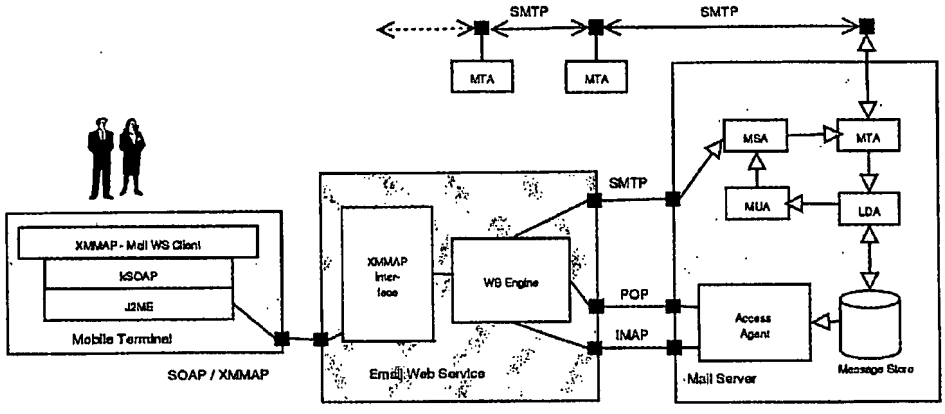
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declaration under Rule 4.17:
— *of inventorship (Rule 4.17(iv)) for US only*

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: A METHOD, PROTOCOL FORMAT AND SYSTEM FOR MOBILE EMAIL COMMUNICATION



(57) Abstract: A system, method and protocol are disclosed for transferring messages to and from a mobile terminal from an email server. The emails are transferred by means of an email service server with a SOAP interface towards the mobile terminal and the common POP/IMAP/SMTP interfaces towards the email server.

WO 2005/117372 A1

A METHOD, PROTOCOL FORMAT AND SYSTEM FOR MOBILE EMAIL COMMUNICATION

Technical field

The present invention relates to the fields of mobile communication and Internet mail, and in particular relates to a method, protocol and system for transferring emails to and from a mobile device.

Technical background

Current mail communication systems consist of an email server for sending and receiving mail through the Internet, and an email client for reading and writing mail. The client is normally residing on a personal computer with ample processing power and storage capacity. Mail transfer is a complicated process making use of several communication protocols. The protocols are designed with a fixed terminal in view, which create problems when a user desires to read or send mail from a terminal with more limited capabilities, e.g. a mobile phone. Existing solutions do not take into account the limitations of the mobile phones, i.e. limited processing and storage capabilities, limited navigation capability, small keypads. Neither the limitations of the wireless networks, i.e. unstable conditions and high latency, nor unpredictable QoS are dealt with. Specifically, the problems that arise are due to the protocols creating excessive message exchange between client and server and unnecessary overhead in different parts of the email. This creates problems in a communication channel of limited bandwidth. In addition, too complex representations of the emails create problems for the presentation on a display screen of limited size and resolution. Other problems are due to unsupported presentation formats. A mobile terminal has normally both limited processing power and capacity for storing programs. Email clients are often of a complex nature, which are heavy to implement on a mobile client with limited capabilities. Lastly, strict firewall configurations may create problems for mobile access.

There has been attempts on creating more efficient solutions for mobile mail access, e.g. by employing VPN channels between the server and the client, or by introducing an additional Web/WAP mail server handling the traffic between the email server and the client. However, these solutions create problems on their own.

Brief description of the invention

Thus, there is a need for a solution for mobile access to emails that obliterate the drawbacks with prior art mail communication systems enumerated above.

It is an object of the present invention to provide a system and protocol for mobile email communication that is better suited for a communication channel of limited bandwidth than prior art systems.

Another object is to provide a system and protocol by which the emails can be properly presented on a display screen of limited size.

Still another object is to provide a system and protocol that is less demanding as to processing power and storage capabilities in the mobile client.

The objects above are achieved in a system for mobile email communication as claimed in the appended patent claim 1, a corresponding method as claimed in claim 3, as well as a protocol for the transfer of email messages as claimed in claim 18. Preferred embodiments of the invention appear from the matching dependent claims.

Specifically, according to a first aspect, the invention consists of a method for mobile email communication between an email server with POP/IMAP/SMTP interfaces and a mobile terminal with an email client with a SOAP interface through an email services server with POP/IMAP/SMTP interfaces, as well as a SOAP interface, said method including:
sending SOAP message requests from the mobile terminal to

the email services server, containing predefined method calls having as parameter a XML protocol format message, converting the requests in the email service server into standard email messages, and sending said standard email messages to the email server and *vice versa*.

According to a second aspect, the invention consists of an email communication protocol format for messages to be transferred to and from an mobile terminal, including:
a <Headers> element containing a limited set of information relevant for a user of the mobile terminal,
a <Flags> element mapped directly from the IMAP specification,
a <Body> element of "text/plain"-MIME type without alternative representations.

According to a third aspect, the invention consists of a system for mobile email communication, said sytem including:
an email server with POP/IMAP/SMTP interfaces,
a mobile terminal with an email client with a SOAP interface,
an email services server with POP/IMAP/SMTP interfaces, as well as a first SOAP interface,
said email services server being arranged to interpret XML protocol format message requests from the mobile terminal received on said first SOAP interface, convert said message requests into POP/IMAP/SMTP format messages, send the converted messages to the email server, convert the result into SOAP messages and send the SOAP messages too the mobile client.

Brief description of the drawings

The invention will now be described in detail in reference to the appended drawings, in which:

Figure 1 is showing the current mail system architecture (prior art),

Figure 2 is showing the relationship between an SMTP mail object and a Internet Message Format message (prior art),

Figure 3 is showing an SMTP - envelope (prior art),

Figure 4 is showing examples of email structures (prior art),

Figure 5 is showing a VPN solution (prior art),

Figure 6 is showing a Web/WAP Mail overall architecture (prior art),

Figure 7 is showing a proposed new solution with straightforward SMTP, POP, IMAP mapping to XML Web Service,

Figure 8 is showing an alternative new solution with the XML Web Service taking use of XMTP,

Figure 9 is illustrating the solution of the invention sending an e-mail using XMMAP,

Figure 10 shows the inventive email Web Service Architecture using XMMAP,

Figure 11 is showing the messaging occurring between participating components when sending an e-mail using XMMAP,

Figure 12 shows the Web Service Interfaces,

Figure 13 shows a Web Service Enterprise model.

Detailed description

As shown in Figure 1, the current mail communication systems consist of two main components, email Server and email Client. They are both compositions of several elements that are making use of several communication protocols, as well as internal service elements. Examples of protocols are SMTP (Simple Mail Transfer Protocol) [4],

IMAP (Internet Message Access Protocol) [5] and POP (Post Office Protocol) [6].

To send a mail, the user interacts with the UI (User Interface), which allows her to compose an email. When the user chooses to send the email, the UI will hand over the message to the MUA (Mail User Agent) that will establish an SMTP session with the remote MSA (Mail Submission Agent) to expedite the mail. The MSA can do pre-specified adaptations to the message to comply with the SMTP-IMF standards [7]. Next it delivers the mail either to a local user through a LDA (Local Delivery Agent), or passes it on to an MTA (Message Transfer Agent) which relays the mail to it's final recipient(s).

An SMTP mail object contains an envelope and content. The SMTP envelope is sent as a series of SMTP protocol units. It consists of an originator address (to which error reports should be directed); one or more recipient addresses and optional protocol extension material. Historically, variations on the recipient address specification command (RCPT TO could be used to specify alternate delivery modes, such as immediate display; those variations have now been depreciated

The SMTP content is sent in the SMTP DATA protocol unit and has two parts: the headers and the body. If the content conforms to other contemporary standards, the headers form a collection of field/value pairs structured according to the Internet Message Format; the body, if structured, is defined according to MIME (Multipurpose Internet Mail Extensions).

The mail will then be sent from MTA to MTA and will finally arrive to the final delivery MTA that deposits the mail in the Message Store through an LDA. This concludes the SMTP transfer of the message.

In order to send and/or relay mail, we must follow the protocol described in RFC2821 [4]. This sequence of commands is often referenced to as the "SMTP Envelope".

SMTP Envelope – excessive message transfers

This is an example of a typical "SMTP Envelope" for sending a normal e-mail to two recipients. It's worth noting the excessive message transfers:

```
220 dus12.nta.no ESMTP Exim 3.35 #1 Fri, 09 Apr 2004 15:57:01 +0200
EHLO dus12.nta.no
250-dus12.nta.no Hello jonfinng at localhost [127.0.0.1]
250-SIZE
250-PIPELINING
250 HELP
MAIL FROM:<jonfinng@stud.ntnu.no>
250 <jonfinng@stud.ntnu.no> is syntactically correct
RCPT TO:<luser@dus12.nta.no>
250 <luser@dus12.nta.no> verified
RCPT TO:<jonfinng@stud.ntnu.no>
250 <jonfinng@stud.ntnu.no> verified
RCPT TO:<jonfinng@dus12.nta.no>
250 <jonfinng@dus12.nta.no> verified
DATA
354 Enter message, ending with "." on a line by itself
X-header: Testmessage
Headers (header part of the message) goes here…
content (body part of the message) goes here…
attachments goes here…
.
250 OK id=1BBwZ4-0000nZ-00
QUIT
221 dus12.nta.no closing connection
```

As seen in Figure 3 a minimum of 11 message transfers are needed between the client and server in order to send a single mail. Two extra transfers are introduced for every additional recipient. When sending multiple mails using the same connection to the mail server, a minimum of nine message transfers is needed. This is undesirable, it is a goal to keep the number of message transfers to an absolute minimum. Since message exchanges introduce extra overhead and delay, which is especially important to keep at a minimum when using a wireless link with low bandwidth.

IMF and MIME – Unnecessary Headers and Complex Body Structures

IMF (Internet Message Format) and MIME (Multipurpose Internet Mail Extensions) [8] are standards used for representing the actual content of the email. IMF defines which headers are mandatory, and how a standard message should be organized. MIME is an extension to IMF, which defines how complex emails should be represented. This is typically emails which has file-attachments, multiple alternative representation formats or even enclosed email-messages within the message itself.

The IMF representation of an email is quite efficient, and does not introduce any overhead of significance. This makes IMF a good starting point when trying to represent emails adapted for mobile terminals. The problem with IMF is not the format itself, but how it is used. An email usually contains a lot of information enclosed in headers. Most of this information is not relevant to the end user. This information may include a list of mail servers the message has visited on its way to its destination, and several so called "X-headers"1. This information is usually not presented in the user-agents and introduces a significant amount of overhead when sending email headers to mobile agents.

This is an example of the headers in a typical email message:

```
From thanh-van.do@telenor.com  Sat Apr  3 15:37:46 2004
Return-Path: <thanh-van.do@telenor.com>
X-Original-To: jonfinng@homeo.stud.ntnu.no
Delivered-To: jonfinng@homeo.stud.ntnu.no
Received: from merke.itea.ntnu.no (merke.itea.ntnu.no [129.241.7.61])
        by bison.stud.ntnu.no (Postfix) with ESMTP id 214FE321
        for <jonfinng@homeo.stud.ntnu.no>; Sat,  3 Apr 2004 15:37:46 +0200
(MEST)
Received: by merke.itea.ntnu.no (Postfix)
        id 1151913DA83; Sat,  3 Apr 2004 15:37:46 +0200 (CEST)
Received: from localhost (localhost [127.0.0.1])
        by merke.itea.ntnu.no (Postfix) with ESMTP id DABD013DCAB
```

[1] "X-headers" are custom headers for providing extra information about the email. Typically added by user agents, virus and spam-scanners.

```
        for <jonfinng@stud.ntnu.no>; Sat,  3 Apr 2004 15:37:45 +0200 (CEST)
Received: from virus-out-st.online.no (virus-out.ttyl.com [193.212.240.200])
        by merke.itea.ntnu.no (Postfix) with ESMTP id 5B43C13DB64
        for <jonfinng@stud.ntnu.no>; Sat,  3 Apr 2004 15:37:43 +0200 (CEST)
Received: from tns-fbu-22-209.corp.telenor.no ([134.47.162.190]
[134.47.162.190]) by scan.telenor.net with ESMTP for
jonfinng@stud.ntnu.no; Sat, 3 Apr 2004 15:36:55 +0200
Received: from tns-fbu-22-212.corp.telenor.no ([134.47.162.91]) by tns-fbu-22-
209.corp.telenor.no with Microsoft
SMTPSVC(5.0.2195.6713);
         Sat, 3 Apr 2004 15:36:54 +0200
Received: from tns-fbu-2e-004.corp.telenor.no ([134.47.163.148]) by tns-fbu-
22-212.corp.telenor.no with Microsoft
SMTPSVC(5.0.2195.6713);
         Sat, 3 Apr 2004 15:36:54 +0200
Content-class: urn:content-classes:message
X-MimeOLE: Produced By Microsoft Exchange V6.0.6487.1
Subject: RE: Paper MWCN 2004 final version
Date: Sat, 3 Apr 2004 15:36:53 +0200
Message-Id: <375BFFDA619DBC4AA93996BF1F1D23969CD6BB@TNS-FBU-2E-
004.corp.telenor.no>
X-MS-Has-Attach: yes
X-MS-TNEF-Correlator:
Thread-Topic: Paper MWCN 2004 final version
thread-index: AcQXAvMvvLDODbn9SeSq5/2eImqLIQCfUkcw
From: <thanh-van.do@telenor.com>
To: <jonfinng@stud.ntnu.no>
X-OriginalArrivalTime: 03 Apr 2004 13:36:54.0486 (UTC)
FILETIME=[BA45C760:01C41980]
MIME-Version: 1.0
Content-Type: multipart/mixed;
  boundary="----_=_NextPart_001_01C41980.B9E57E06"
X-Content-Scanned: with sophos and spamassassin at mailgw.ntnu.no.
X-Spam-Level:
X-Spam-Status: No, hits=1.0 required=3.0 tests=__ORIG_MESSAGE_LINE
version=2.20
X-Spam-Level: *
Content-Length: 140839
Lines: 1859
Status: RO
X-Status: A
X-Keywords:
X-UID: 5466
```

The headers marked in bold are the headers usually presented in a user-agent because they contain the information most relevant to the end user. When stripping out all other headers, the size of the header-element of this email shrinks from 2351 to 323 bytes, significantly reducing this emails total size. Hence, for mobile terminals with limited bandwidth and client-functionality, only a minimal set of headers should be transferred.

A MIME email's content is organized in two dispositions: INLINE and ATTACHMENT. The parts marked as INLINE will usually be shown in the mail client when opening the email for reading. The INLINE content parts are encoded as text.

The parts marked as ATTACHMENT consist of binary data, and must be opened in an external application or through a plug-in in the mail reader able to interpret that specific attachment. The different parts of the message are separated by custom defined "boundaries". When a boundary appears in the message, it marks the beginning of a new part. Every part must be identified by a "Content-type" parameter, telling what MIME-type the current part is. Other parameters to a body part may be the encoding and file name of the attachment.

Here is a cut from an email containing a normal text part as INLINE and an attached picture as ATTACHMENT:

```
******* stripped headers ***************
Mime-Version: 1.0
Content-Type: multipart/mixed;
boundary="----=_NextPart_000_2a6d_50c1_182a"
Message-ID: <BAY14-F29YIgoFT3kMU000118b4@hotmail.com>
X-OriginalArrivalTime: 02 Mar 2004 11:54:46.0152 (UTC)
FILETIME=[28483880:01C4004D]
This is a multi-part message in MIME format.
------=_NextPart_000_2a6d_50c1_182a
Content-Type: text/plain; charset=iso-8859-1; format=flowed

_____
MSN Hotmail http://www.hotmail.com
------=_NextPart_000_2a6d_50c1_182a
Content-Type: image/pjpeg; name="teamaa.jpg"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="teamaa.jpg"
/9j/4AAQSkZJRgABAgAAZABkAAD/7AARRHVja3kAAQAEAAAAUAAA/+IMWElD
Q19QUk9GSUxFAAEBAAAMSExpbm8CEAAAbW50clJHQiBYWVogB84AAgAJAAYA
********* stripped binary data ***********
```

The first header element included in this cut tells us that this is a MIME message, and that the mail reader must support MIME to understand this message. When the message contains parts with different content-types the base content type is marked as "multipart/mixed" as in this message. One should as well pay attention to the boundaries in this message (——=_NextPart_000_2a6d_-50c1_182a), separating the two parts of this email.

The first bodypart is of type "text/plain" and represents the text in this message. Parameters tells us that the bodypart has charset iso-8859. The second part of the message is an attached picture with content type "image/pjpeg". In this part, the filename and encoding are supplied as parameters.

This example of how to organize the contents of an email is just one of numerous possible ways a message may be structured. MIME messages also support nested content, and so called "multipart/alternative" body parts (among many other content types). A "multipart/alternative" message holds different presentation of the same data in different formats, allowing the client or user to choose which format that is best suited for presentation. Figure 4 shows some examples of how email may be organized in different bodyparts.

MIME is a good and much needed extension to ordinary mail. But the emails' representations may become very complex, and may take time to process on terminals with limited processing capabilities. MIME requires that the end clients have support for showing or opening the attachments sent. In addition, there is no point sending multiple representations of the message itself (multipart/alternative) if the client cannot present it to the user. In such cases, MIME body parts will end up as unnecessary overhead.

Access Agents and Firewalls

The access and manipulation of emails stored in the Message Store are enabled by an Access Agent, which implements protocols such as IMAP (Internet Message Access Protocol) or POP (Post Office Protocol). The client application will contain either an IMAP client or a POP client (or both) for the retrieval of emails and mailbox operations. This of course, comes in addition to the software required for sending out mail using SMTP.

In many cases the clients and server are on an Intranet protected by a corporate firewall, which prevents access to the email server from outside this network. This is done to protect the server from outside attacks and hacker attempts, as well as for stopping unwanted relaying activity, which may include spamming. On the other hand, users are getting more and more mobile and want to access their email accounts also when not being on the corporate intranet. This introduces a conflict interest between security and availability.

The problems with today's mail systems when accessed by mobile terminals can be summarized as follows:

Excessive message exchange.

Unnecessary overhead in different parts of the email.

Too complex representations of the emails.

Unsupported presentation formats.

Complex implementation of email clients. Different protocol implementations are needed for sending and retrieving mail.

Problems related to mobile access, due to strict firewall configurations.

Current work-arounds and their limitations

Much effort has been put into solving the problem related to firewall configurations, often denying access to mail operations outside the corporate intranet. Several more or less successful attempts have emerged. The solutions might work well with laptops and stationary computers, but so far, the solutions have severe limitations when been used for providing email to mobile phones.

Virtual Private Network

In this solution, shown in Figure 5, a VPN client establishes a secured channel from the remote computer through the Internet and across the firewall to the corporate intranet. The client is hence logically connected to the intranet and the user can use a normal email client to access his email using standard protocols.

This solution is very suitable for a PC with sufficient processing and storage capabilities and a fairly stable connection with considerable transfer rate. This solution does not fit for mobile phones because of several issues as follows:

Most of mobile phones are not equipped with a VPN client.

Mobile phones do not have the sufficient processing capability to equalize the overhead introduced by the encryption and decryption.

The unstable wireless link may introduce problems to the VPN session.

Sporadic mail access by the mobile user is not suitable because of long setup time and possible VPN connection timeout.

Mobile phones may not have sufficient storage for all the mails.

Mobile phones may not have processing and storage capabilities to host both a SMTP client and an IMAP/POP client.

Mobile phones have much more limited User Interface capabilities, i.e. small display, limited navigation facility, and small and limited keypad.

Mobile phones come in a variety of types making the design of user interface difficult.

Web/WAP mail

As shown in Figure 6 , a Web/WAP mail Server is now introduced. It is usually resides in the DMZ (De-Militarized Zone) between the Internet and the corporate intranet. The Web-application server synchronizes with the corporate mail server behind the firewall for the retrieval and sending of mail. Actually, this server contains a Web application with same core elements as an email client. Both implementations use SMTP and IMAP/POP for communication with the email server. The user can access her mails using a WWW-browser.

The main advantage of this solution is that a mobile user needs nothing more than a WWW-browser to access her email. No additional functionality is required on a mobile phone. The mobile phone can access the user's email account on the webmail server either directly using an HTML/xHTML browser or via WAP using a WML browser. It is worth noting that this alternative applies also for pure web mail services like hotmail, yahoo, online, etc.

The disadvantages are:

The mails are not stored on the mobile phones but on the Web/WAP Mail server. To read the same mail twice, the user has to access the Web/WAP server again. This due to the relaxed HTML/WML-standard for which it is difficult to parse out the actual information contents from presentation wrapping.

The reading of mail may be time consuming and less flexible since the web/WAP pages are generated dynamically on the fly for each access. Caching on the mobile device is difficult and requires much storage capacity, again due to the excessive amount of presentation data supplied together with the information content.

The mails are not adapted displaying in small displays.

Problems with known solutions

As discussed earlier, the existing solutions for providing mobile access to emails suffer from the fact that they did not take into account the limitations of the mobile phones, i.e. limited processing and storage capabilities, small display, limited navigation capability, small keypads. Neither the limitations of the wireless networks, i.e. unstable connections and high latency, nor unpredictable QoS are properly dealt with.

The invention

This invention consists of two elements:

An architecture based on XMMAP that enables the mobile user to access, send and manipulate her mails stored in a mail server residing in the user's corporate intranet.

A protocol called XMMAP (XML Mobile Email Access Protocol) consisting of a set of methods and a data model that alleviates the functional requirements of the mobile phones, and reduces the amount of data sent to the mobile phones.

Mobile mail access with SMTP and IMAP/POP Web Service

To enable the mail access from mobile phones, the XML Web Service concept is found most suitable due to the ubiquity of the World Wide Web and the ability to traverse firewalls using SOAP (Simple Object Access Protocol) [2]. The most straightforward solution is to expose the whole SMTP and IMAP/POP as Web Services as shown in Figure 7. Each SMTP and IMAP/POP command is mapped to a Web Service method. In fact, each SMTP or IMAP/POP command is encapsulated in a SOAP message and transported to the WS client.

An advantage in this solution compared to Web/WAP solutions is that there are no overhead data specifying the appearance of the content. Compared to VPN, this solution is more robust due to the use of SOAP. SOAP has a more relaxed way to handle sessions. This involves that sessions

may survive even if the link is goes down for a period of time.

The disadvantage is the numerous functional requirements that are put on the mobile phone. To access and retrieve mails, the WS client must be capable of understanding the SMTP/IMAP/POP commands and communicating properly with the email Server. It must therefore be equipped both with a MUA (Mail User Agent) with full SMTP support as well as an IMAP/POP client. This functionality is put on top of the SOAP engine.

Other disadvantages are the high number of interactions and also the high amount of downloaded data that are an implication when implanting this solution. This is definitely not suitable in wireless network with limited bandwidth.

IMAP and SMTP are not pure request-response protocols. They also include the possibility of server-originated messages. If full compliance with SMTP and POP/IMAP should be achieved, the clients must be able to listen for method invocations from the server. In this case server originated method invocation implies a full Web Service implementation on the client. This is infeasible due to limited processing and storage capacity on mobile terminals. They do not have a static internet address, and the connection is frequently shut down. In other words, mobile phones were not designed to work as servers.

Mobile Mail Access with XMTP

The use of XML Web Services is taken a bit further by the introduction of XMTP (XML MIME Transformation Protocol) [1]. XMTP is a protocol for mapping IMF or MIME messages to an XML representation.

When using XMTP, we are able to transport the entire message in XML (see Figure 8). This makes it easier for the client to parse the message. An XDS (XML Definition Schema)

can be used for interpreting the different fields of the message, making presentation of the mail in the client potentially faster.

XMTP contains no functionality beyond the IMF message mapping. This means that when using XMTP, all message exchanges and overhead must remain as in the previous approach. XMTP may in itself introduce some amount of overhead due to the XML-tags.

There is very little gain using XMTP instead of direct mapping, it even adds more complexity on the server side by introducing a conversion routine between SMTP and XMTP. An approach using XMTP is therefore turned down as suitable for usage with mobile terminals.

Mobile Mail Access with XMMAP

The goal of the introduction of XMMAP (XML Mobile Email Access Protocol) is to solve some of the major problems related to e-mail access from mobile terminals, and overcome the limitations and insufficiencies of the previously proposed solutions.

As we could see from the message sequence chart in Figure 3, a minimum of 11 messages are required between the client and server just in order to send an email. This is highly undesirable when using a mobile terminal. If the link goes down in the middle of a message sequence, the entire procedure has to be repeated. Additionally, every message transfer introduces unnecessary overhead from underlying protocols. On top of this we have the considerations regarding overhead in headers and representation as discussed earlier.

By introducing XMMAP we reduce the number of message transfers to two for most IMAP/POP/SMTP operations. As shown in Figure 9, we are down to one message from the mobile client to the web service, and one message in return.

POP and IMAP do not have as many message transfers as SMTP. Often only one request and response requiring is required per invocation. On the other hand, these protocols require that the user authenticates himself before invoking any operation on the account. This introduces session management and extra delay and overhead.

The major benefit when using XMMAP is that it both combines and simplifies the functionality and information given by the MTA (SMTP), Access Agent (IMAP/POP) and the representation format (SMTP-IMF/XMTP). This is achieved by mapping the information into an XML format and tying different parts of the format to specific requests and responses to SOAP - methods.

As XMTP, XMMAP utilizes the strength of XML to simplify parsing of the received information. The terminal can use an XML-parser for retrieving the information from the XMMAP-message as well as for constructing new XML documents. This makes implementation of an XMMAP-compatible mail client a very simple task compared to a full scale email client supporting SMTP with a wide range of MIME-types, as well as the client implementations of POP and IMAP.

XMMAP introduces a new concept of messaging. While traditional protocols rely on frequent message transfers with well-defined operations, XMMAP is more flexible, making it possible to do most operations in one exchange.

XMMAP is in its basic form a representation of an entire mail account, spanning everything from login credentials to flags in a specific message. This makes it possible to use sub-parts of the XMMAP-format for representing different parts of a mail account for different purposes. This is especially useful when utilizing XMMAP for invoking methods on the mail server. When invoking a method, only the relevant subparts of the formats are sent, thus avoiding unnecessary overhead.

XMMAP data format

As mentioned, XMMAP is a representation of an entire mail account. This section holds a brief description of the format including several examples. We begin straight at the point by giving an example of how the most important part of an account, a single message itself, is represented in XMMAP:

```
<Message
    xmlns=''http://www.finngard.org/2004/03/xmmap/''
    xmlns:web=''http://www.w3.org/1999/02/22-rdf-syntax-ns#''
    web:about=''mid:10784063170022320lycos-europe.com''>
    <BoxNumber mailbox=''INBOX''>12</BoxNumber>
    <Headers>
        <From>John Doe &lt;johndoe@telenor.com&gt;</From>
        <To>&lt;finngard@finngard.org&gt;</To>
        <Subject>XMMAP protocol, suitable for PDA?</Subject>
        <Date>Fri, 2 Mar 2004 12:23:12 -0700</Date>
    </Headers>
    <Flags protocol=''imap''>
       <Seen>1</Seen>
       <Answered>0</Answered>
    </Flags>
    <Body charset=''ISO-8859-1''>
      Hi! Do you suggest using the XMMAP format for PDA's as well as
      mobile phones?
    </Body>
    <Attachments>
       <Attachment content-type=''application/x-ms-word''
            encoding=''base64''>
         <AttachmentNumber>1</AttachmentNumber>
         <Filename>pda_description.doc</Filename>
         <Size>1345</Size>
         <Content>/9j/4AAQSkZJRgABAQEAYABgAAD//gAcU29mdHdhcmU6IE1pY3
                  Jvc29mdCBPZmZpY2X/2wBDAAoHBwgHBgoICAgLCgoLDhgQDg
                  0NDh0VFhEYIx8lJCIfIiEmKzcvJik0KSEiMEExNDk7Pj4+JS
                  5ESUM8
                ....
         </Content>
       </Attachment>
    </Attachments>
</Message>
```

The format is very much self-explanatory as it contains elements from the mentioned known protocols (SMTP, XMTP, POP3 and IMAP). XMMAP maps perfectly into a standard SMTP-IMF message, with or without MIME extensions, and vice versa. In addition to this, an XMMAP-message supplies the essential information given by both POP3 and IMAP mail access protocols.

The <Message> element is the root of a MIME/RFC2822 message. A <Message> element contains <BoxNumber>, <Headers>, <Flags>, <Body> and <Attachments> elements.

Explanation of elements in XMMAP

namespaces

The namespace links to a dated URI, which holds relevant XML Schemas for describing the current usage of the XMMAP protocol.

web:about attribute

This attribute is adopted from the XMTP-format, and contains the message identifier URI.

<BoxNumber> element

A <BoxNumber> represents the message's number in a mailbox. The actual mailbox may be given as parameter when not implicit from surrounding context.

<Headers> element

The <Headers> contains standard RFC 2822-headers with local names representing the header names. Parameters are represented by child elements.

One major point when using XMMAP is to not transfer each and every header contained in the original SMTP-IMF-message. Instead it should be aimed at using only a limited set of headers transferring information relevant to the end user. E.g. an SMTP message often contains several ``Received'' and ``X'' headers. This information might be relevant when backtracking message-paths and for system specific processing tasks (such as firewalling and spam filtering). But this information is not relevant to the end user. To save storage space and transmission capacity on

the mobile terminal, as many header elements as possible should be stripped from the message. This is done by the SMTP->XMMAP gateway before delivering the message to the mobile terminal.

<Flags> element

The <Flags> element supplies the current flags of an email as child elements. When retrieving previously set flags with XMMAP, the set flags are transmitted and the tag holds the value "1". When setting flags, a value of "1" means that the flag should be set, and "0" means that the flag should be unset. The flags are mapped directly from the IMAP specification.

<Body> element

The body element contains the body of the email message. The current character set is given as a parameter. If no parameter is given ``us-ascii'' is assumed as in SMTP-IMF. The content of the <Body> element is always represented as "text/plain"-MIME type. If the original message supplies optional ways of representing the message, all but one alternative are stripped from the message by the SMTP->XMMAP gateway. If the body is only represented in for example "text/html", the HTML-tags are stripped. The results after stripping are supplied as "text/plain". This is done to reduce the amount of data to transmit, as well as supplying the data in a format as simple as possible for use with mobile terminals.

Optionally the representation alternatives can be supplied and the corresponding MIME types given as parameters to the <Body> element. If the client cannot represent a body part, an error message will be supplied and shown to the user.

Example of alternative representations of <Body>-element:

```
<Body charset=''ISO-8859-1''>
   MESSAGE COULD NOT BE REPRESENTED IN TEXT/PLAIN!
</Body>
<Body charset=''ISO-8859-1'' content-type=''text/xml''>
```

```
    &lt;xml&gt;
    &lt;Element&gt;Data&lt;/Element&gt;
    ....
</Body>
```

<Attachments> element

Only relevant if the message contains attachments. These are supplied within the Attachments-element. Each attachment is contained within an Attachment-element. The MIME type and encoding is given as attributes. Encoding may be omitted, base64 is then assumed. In addition, an <Attachment> element contains <Filename>, <Size> and <Content> elements, containing the filename, size of the file and binary encoded data, respectively.

Attachments may be large, and in many cases the user is not interested in downloading these, but only receive the message-text and attachment-description. In such cases the <Content> element may be left empty (<Content/>). Even if the <Content> element is empty, the user receives information about the filetype, filename and size of the file. From that information the user can decide whether he wants to download the attachments later or not.

The described elements are sufficient for representing an email message. Including elements both from SMTP-IMF representation format as well as flag information used in access protocols. But in order to represent an entire account, some additional elements are needed. A representation of an entire account using XMMAP are shown below.

Account representation using XMMAP:

```
<Account xmlns=''http://www.finngard.org/xmmap/''>
  <UserName>jonfinng</UserName>
  <PassWord>l337s3cr37</PassWord>
  <Protocol>IMAPS</Protocol>
  <Port>443</Port>
  <MailBoxes>
     <MailBox>
        <BoxName>INBOX</BoxName>
        <Unread>12</Unread>
        <Total>23</Total>
        <Messages />
```

```
            <MailBoxes>
               <MailBox>
                  <BoxName>INBOX.old</BoxName>
                  <Unread>0</Unread>
                  <Total>10</Total>
                  <Messages />
                  <MailBoxes />
               </MailBox>
            </MailBoxes>
         </MailBox>
         <MailBox>
            <BoxName>Work</BoxName>
            <Unread>3</Unread>
            <Total>34</Total>
            <Messages>
               <Message xmlns=''http://www.finngard.org/2004/04/xmmap/''
                  xmlns:web=
                      ''http://www.w3.org/1999/02/22-rdf-syntax-ns#''
                  web:about=''mid:1078406317002232@lycos-europe.com''>
                  ...
               </Message>
            </Messages>
            <Mailboxes />
         </MailBox>
      </MailBoxes>
   </Account>
```

<Account> element

The Account element represents an entire mail account with all its meta-information as well as content. The account contains <Mailboxes>-elements. Account elements can also include login credentials. This can be <UserName>, <PassWord>, <Protocol> and <Port>. The <Account> element is necessary only when logging on to an account, and when receiving a list of available mailboxes from the server.

<Mailboxes> element

An account may contain one or more mailboxes; these are listed within the Mailboxes-element-tag as MailBox-elements.

<MailBox> element

Information about how many messages a mailbox contains, and how many of them that are new messages is often wanted. One way of retrieving this information is to download all messages in a box and check which one of them that has the

right flags set. This is not clever to do on mobile terminals with limited storage and processing capacity. IMAP and POP have functionality that gives you this information without needing to download all messages.

The Mailbox-element in the XMMAP-format offers information usually provided by POP and IMAP. The element can also hold full representation of Mailboxes with content. <BoxName>, <Unread>, <Total>, <Messages> and <Mailboxes> are sub elements of a Mailbox-element. <BoxName>, <Unread> and <Total> contains the name of the mailbox, the number of unread messages in it and the total number of messages respectively.

<Messages> element

The Messages-element can contain one or more of the messages of this folder. It can also be left as an empty tag if this information is not wanted in the current request. A Mailboxes-element within a MailBox-element contains sub-mailboxes of the current mailbox if such exists.

XMMAP defined methods

The XMMAP data format is useful for representing an account in a minimalist way, and may also be well suited for storage purposes. On the other hand, the data format lacks the coupling to specific methods related to mobile mail access. This coupling is achieved by defining SOAP - methods using XMMAP messages as parameters.

The following set of methods is implemented so far:

loginMobileTermXMMAP	LoginProfileXMMAP
getMailboxesXMMAP	getHeadersAsXMMAP
getNewHeadersAsXMMAP	getMessagesAsXMMAP
setFlagXMMAP	deleteXMMAP

sendMailXMMAP	logoutXMMAP

Table 1 XMMAP methods

Method List

loginMobileTermXMMAP and loginProfileXMMAP

These are methods used for authentication. Mandatory for retrieval of mail, since remote IMAP and POP servers requires authentication. It is recommended to use these methods for authentication regardless of proceeding operation. This is because authentication of users largely prevents misusage of the service, as well at it offers better back tracing possibilities when undesired usage is detected.

When invoking one of these two methods a session is established by the Web Service. It is recommended that this session is a SOAP session, but may also be based on underlying protocols such as HTTP. The Web Service also manages a connection to the mailserver related to each session. The session fetches a previously stored profile, or creates a new one. The profile, among other information, holds custom adaptation properties for the mobile terminal used in this session.

The difference between these two methods is that "loginProfileXMMAP" uses a pre stored profile, while "loginMobileTermXMMAP" takes more input parameters which are needed for creating a new profile for present and future use. These parameters are typically related to the mobile terminal as screen size and maximum number of colors supported. These parameters are not a part of the XMMAP-message itself.

The response of an invocation of these methods is a session- and profile ID in addition the XMMAP-message (see example below). The XMMAP message is a list of the mailboxes in the requested account.

Request message example:

```
<Account>
      <UserName>jonfinng</UserName>
      <PassWord>secret</PassWord>
      <Host>imap.stud.ntnu.no</Host>
      <Protocol>IMAPS</Protocol>
      <Port>443</Port>
</Account>
```

Response message example:

```
<Mailboxes>
      <Mailbox>
            <BoxName>INBOX</BoxName>
            <Unread>2</Unread>
            <Total>23</Total>
      </MailBox>
      <MailBox>
            <BoxName>INBOX.old</BoxName>
            <Unread>0</Unread>
            <Total>4</Total>
      </MailBox>
</Mailboxes>
```

getMailBoxes

This method retrieves the mailboxes from an account when already logged in. The message exchange is quite similar to the login-methods. The request can however be stripped down to <Account /> or completely omitted since the user is already logged in, and the account information is stored within the session.

getHeadersAsXMMAP and getNewHeadersAsXMMAP

These methods are used for fetching headers from messages in a mailbox. It is often desirable to only fetch information about unread mail to avoid flooding the screen on the mobile terminal with information about old emails. Therefore, a method called "getNewHeadersAsXMMAP" should be implemented in addition to "getHeadersAsXMMAP". Which

headers to send and which to strip off must be set as a part of the user profile.

Request message example:

```
<Mailbox>
      <BoxName>INBOX.old</BoxName>
</Mailbox>
```

Response message example:

```
<Messages>
      <Message web:about=''mid:10784063170022232@lycos-europe.com''>
            <Headers>
                  <To>me@here.com</To>
                  <From>someone@there.com</From>
                  ...
            </Headers>
      </Message>
      <Message>
            ...
      </Message>
</Messages>
```

getMessagesAsXMMAP

As the name suggests, this is the message for retrieving mail message content. The request message must provide the name of the mailbox and message number(s). Note that the content of the eventual attachments are not initially sent, but only the information describing them. The user can then choose to download them later using the "getAttachmentsAsXMMAP" method.

Request message example:

```
<Mailbox>
      <BoxName>INBOX</BoxName>
      <Messages>
            <Message>
                  <BoxNumber>12</BoxNumber>
            </Message>
            <Message>
                  <BoxNumber>21</BoxNumber>
            </Message>
      </Messages>
</Mailbox>
```

Response message example:

```
<Messages>
      <Message>
            <Body>
                  ...
            </Body>
            <Attachments>
                  <Attachment
                        content-type=''application/x-ms-word''
                        encoding=''base64''>
                        <AttachmentNumber>1</AttachmentNumber>
                        <FileName>pda_spec.doc</FileName>
                        <Size>1240211</Size>
                  </Attachment>
                  <Attachment
                        content-type=''application/pgp-signature''
                        encoding=''base64''>
                        <AttachmentNumber>2</AttachmentNumber>
                        <FileName>signature</FileName>
                        <Size>202</Size>
                  </Attachment>
            </Attachments>
      </Message>
</Messages>
```

getAttachmentAsXMMAP

This method can be invoked to get the content of an attachment. The client will have received a description of the attachment(s) from a call to the getMessagesAsXMMAP method, and uses the attachment number(s) to identify which attachment(s) within which message(s) it wants. Only the attachment(s) will be returned, the body content is omitted.

Request message example:

```
<Mailbox>
      <BoxName>INBOX</BoxName>
      <Messages>
            <Message>
                  <BoxNumber>12</BoxNumber>
                  <Attachments>
                        <Attachment>
                              <AttachmentNumber>1</AttachmentNumber>
                        </Attachment>
                        <Attachment>
                              <AttachmentNumber>4</AttachmentNumber>
                        </Attachment>
                  </Attachments>
            </Message>
      </Messages>
</Mailbox>
```

Response message example:

```
<Messages>
      <Message>
            <BoxNumber>12</BoxNumber>
            <Body/>
            <Attachments>
                  <Attachment
                        content-type=''application/x-ms-word''
                        encoding=''base64''>
                        <AttachmentNumber>1</AttachmentNumber>
                        <FileName>pda_spec.doc</FileName>
                        <Size>1240211</Size>
                        <Content>/9j/4AAQSkZJRgABAQEAYABgAAD//gAcU29m
                              dHdhcmU6IE1pY3Jvc29mdCBPZmZpY2X/2wBDAAo
                              HBwgHBgoICAgLCgoLDhgQDg0NDh0VFhEYIx8lJC
                              IfIiEmKzcvJik0KSEiMEExNDk7Pj4+JS5ESUM8
                              ...
                        </Content>
                  </Attachment>
                  <Attachment
                        content-type=''image/jpeg''
                        encoding=''base64''>
                        <AttachmentNumber>4</AttachNumber>
                        <Content>
                        ...
                        </Content>
                  </Attachment>
            </Attachments>
      </Message>
</Messages>
```

setFlags

This method can be used to set message flags on the IMAP/POP server. The method returns nothing, except from error messages if something goes wrong. Note: setting the

DELETED flag with setFlags marks the message as deleted, but keeps it in the mailbox. For permanent deletion, see the delete method.

Request message example:

```
<Mailbox>
      <BoxName>INBOX</BoxName>
      <Messages>
            <Message>
                  <BoxNumber>12</BoxNumber>
                  <Flags>
                        <Seen>1</Seen>
                        <Answered>1</Answered>
                  </Flags>
            </Message>
            <Message>
                  <BoxNumber>4</BoxNumber>
                  <Flags>
                        <Deleted>1</Deleted>
                  </Flags>
            </Message>
      </Messages>
</Mailbox>
```

delete

This SOAP method marks a message as deleted, or deletes it permanently (also called expunging) from the IMAP/POP server. Whether or not to expunge is given as a parameter to the method call. The default is not to expunge. The method returns nothing, except from error messages if something goes wrong.

Request message example:

```
<Mailbox>
      <BoxName>INBOX</BoxName>
      <Messages>
            <Message>
                  <BoxNumber>14</BoxNumber>
            </Message>
            <Message>
                  <BoxNumber>15</BoxNumber>
            </Message>
      </Messages>
</Mailbox>
```

sendMail

This is the method to use for sending mail from the mobile client. It is sufficient with one message exchange compared

to SMTP's minumum of 11 exchanges. If using the IMAP protocol, the Web Service sends the final message to the users "sent" box and marks messages as "Answered" in the current mailbox if the message is a reply (given by a 'ReplyNumber' element together with the headers). Setting several <To>, <Cc> or <Bcc> headers in the request message adds multiple recipients in XMMAP. These headers are parsed by the Web Service and converted to SMTP on its outgoing interface. A RFC2822 compliant message are also created by the Web Service from the information received before it is sent to its final destination by SMTP (see Figure 11 is showing the messaging occurring between participating components when sending an e-mail using XMMAP).

Request message example:

```
<Message>
      <Headers>
            <ReplyNumber>12</ReplyNumber>
            <From>Mobile Finngard &gt;finngard@xmmap.com&lt;</From>
            <To>Jomar Jalla &gt;jomar@home.com&lt;</To>
            <To>Britt Jalla &gt;britt@home.com&lt;</To>
            <Subject>Hold yer horses, picture coming up</Subject>
      </Headers>
      <Body>
            Here it is.. :)
      </Body>
      <Attachments>
            <Attachment content-type=''image/png''
                        encoding=''base64''>>
                  <Filename>onthebeach.png</Filename>
                  <Size>12343</Size>
                  <Content>
                        /9j/4AAQSkZJRgABAQEAYABgAAD//gAcU29m
                        dHdhcmU6IE1pY3Jvc29mdCBPZmZpY2X/2wBDAAo
                        HBwgHBgoICAgLCgoLDhgQDg0NDh0VFhEYIx8lJC
                        IfIiEmKzcvJik0KSEiMEExNDk7Pj4+JS5ESUM8
                        ...
                  </Content>
            </Attachment>
      </Attachments>
</Message>
```

This message does not need any response if the Web Service successfully could send the mail further to its final destination. A SOAP-fault describing the error is returned if the mail for some reason could not be delivered.

logout

To log out, the server needs no more information than the session id. Hence the logout call is a SOAP message with empty body.

This will close any connections to the users mail server and delete the session from on the web service server.

Summary

The methods briefly described here is only a limited subset of the most important methods that can be implemented by coupling SOAP and XMMAP for email access on mobile terminals. There is in principle no limitation in what additional methods that may be implemented. The XMMAP format is flexible, and can be extended by new elements in any part of the format if found convenient. There is no fixed order in how the current messages are sent after the user has logged in. When defining new methods, this principle should be followed in order to keep every message independent from both previous and succeeding messages.

Architecture Overview

Figure 10 shows our architecture, and its different software layers and interfaces. The Web Service Client (WS Client) needs support for SOAP and J2ME [3] as with the other proposed solutions. In addition the client this solution only needs a Mail User Agent (MUA) capable of XML document creation and parsing, as well as for sending and receiving XMMAP messages. In other words: No support for any other mail protocols is needed. This makes the client implementation much simpler compared to traditional solutions.

Every operation on the client does only require one XMMAP message sent to the web service, and one message in return. The WS Engine interprets the message client request, does the required communication with the IMAP/POP/SMTP server,

and converts the result into an XMMAP message sent back to the client.

Figure 11 shows the interaction between some of the components from Figure 10 when a mobile client sends an e-mail using the web service. We see how the SMTP interaction is stripped down to one message for the client.

Internal Web Service Functionality

As mentioned briefly earlier, the web service will adapt the mail messages for mobile terminals like cellphones and PDA's. This is achieved by removing unnecessary email headers. Alternative representations on the same content (e.g. both plain text and HTML representations of the message body) is reduced to one. Attachments are by default kept back until the user specifically asks to get them. These actions keep the amount of data transmitted and the number of message exchanges to a minimum.

A summary of the internal functionality offered of the XML Web Service (see Figure 10):

SMTP-IMF to XMMAP gateway and vice versa.

XMMAP to IMAP/POP gateway and vice versa.

Authentication of users.

Holding and managing the user profiles.

Session management for all interfaces and related active connections.

Content adaptation. This includes everything from tag and attachment stripping to picture resizing.

Optional: Local caching of messages and attachments.

Provide necessary interfaces (see Interfaces section).

Interfaces

The XML Web Service should provide these interfaces:

SMTP interface. The XML Web Service will do the necessary communication with the MSA/MTA when sending email. In order to make this work, it has to do all communication with the MTA using the SMTP protocol.

POP/IMAP interfaces. Necessary for retrieving mail, and manipulation mail account. Should also have TSL/SSL support.

SOAP/XMMAP interface. This interface is used for communication with the client, and is thoroughly described in this document.

Optional: SOAP interfaces. One or more interfaces for communication with other XML Web Services. Examples of may be special content filtering services as spam and virus filters.

The XML Web Service should have a fast connection to all interfaces so there will be minimal delays in web service-to-mail server communication. All interfaces should provide support for encrypted communication using SSL/TLS.

Advantages

Our solution gives fewer message transfers between client and server, and less interaction gives better performance for low-bandwidth devices.

Since only the message related information is delivered to the client, messages can be easily stored on the mobile device. This is not feasible when accessing email accounts via e.g. webmail.

The content is automatically adapted to fit the terminal by using information sent by the client software about display dimensions and color support etc. Unnecessary information

is stripped away by the web service before replying to the client.

Session timeout is a problem when working with services requiring authentication. Especially when accessing the Internet via a GPRS network, the GPRS/Internet gateways tend to have a short timeout periods. Reestablishment after a timeout often implies assignment of a different IP address, making all session on higher layers invalid. This problem is solved when using SOAP sessions, ensuring session mobility.

Firewalls are no longer an issue, since all mail traffic can pass through SOAP messages.2

This invention does impose more requirements on mobile phones in terms of processing and storage capabilities.

It is very much aligned with standard technologies such as XML Web Services, IMAP, POP and SMTP.

Limitations and possible improvements

The solution as it stands still has some limitations. Here are some suggestions of how these may be solved:

SOAP and XML introduces overhead itself, eating much of the savings gained when using XMMAP. In order to make the amount of data even smaller, it may be an idea to compress the content of the SOAP message before transferring and decompress it when receiving. However, this will require extra processing capacity on the client. This may potentially imply slow access on clients.

The XMMAP request XML-document may get unnecessary large. For example when only needing to send a mail ID, additional XML-tags are also sent according to the XMMAP standard. A

---

[2] This is of course only true if the corporate firewall opens for global access to a WWW-server (when using HTTP as transport protocol).

good solution to this problem is to offer multiple SOAP methods on the Web Service with same functionality. Equal methods that take either XMMAP or simple SOAP data types as parameters respectively.

There are currently no existing clients supporting XMMAP, which may imply a heavy workload for pioneers implementing the first applications. This of course is valid for everything new, but we believe that the simplicity and flexibility will make XMMAP attractive nevertheless.

Mail retrieval may be slow, since this very much depends on the connection to the remote mail server, which may be slow itself. As long as the connection to the mail server actually is faster than the one to the client, caching is possible. This may be caching of messages when fetching headers, and/or attachments when retrieving the messages. The content will then be preprocessed and ready for transfer when the potential request for it arrives.

Broadening

Our email web service can interact with other web services in an enterprise network of collaborating web services. An example of this is shown in Figure 13.

The e-mail web service can also be configured to work as an MTA, allowing forwarding of e-mails between different instances of the web service.

The web service is not limited to mobile devices, any device with internet access and an XMMAP/SOAP capable client can access it.

Construction of plug-ins utilizing this XML Web Service to standard email user agents as Microsoft Outlook enables global mail access. This service will be as available as Web/WAPmail is today, but enable the user to use his favorite user agent instead of a custom interface through a WWW-browser.

References

[1] XMTP (XML MIME Transformation Protocol)
http://www.openhealth.org/documents/xmtp.htm

[2] SOAP (Simple Object Access Protocol) Specification
http://www.w3.org/TR/soap/

[3] Java 2 Platform, Micro Edition
http://java.sun.com/j2me/

[4] SMTP (Simple Mail Transfer Protocol)
http://www.ietf.org/rfc/rfc2821.txt

[5] IMAP (Internet Message Access Protocol)
http://www.ietf.org/rfc/rfc2060.txt

[6] POP (Post Office Protocol)
http://www.ietf.org/rfc/rfc1939.txt

[7] IMF (Internet Message Format)
http://www.ietf.org/rfc/rfc2822.txt

[8] MIME (Multipurpose Internet Mail Extensions)
http://www.ietf.org/rfc/rfc2045.txt

Claims

1. A method for mobile email communication between an email server with POP/IMAP/SMTP interfaces and a mobile terminal with an email client with a SOAP interface through an email services server with POP/IMAP/SMTP interfaces, as well as a SOAP interface,
characterized in
sending SOAP message requests from the mobile terminal to the email services server, containing predefined method calls having as parameter a XML protocol format message, converting the requests in the email service server into standard email messages, and sending said standard email messages to the email server and *vice versa*.

2. A method as claimed in claim 1, said method including loginMobileTermXMMAP and loginProfileXMMAP method calls for logging in and authenticating the user having as parameters in the XML message at least the user name, the user's password and the user address, optionally also the protocol to be used towards the email server and the port on the email server.

3. A method as claimed in claim 1, said method including a getMailBoxes method call for retrieving a mailbox from an account when logged in.

4. A method as claimed in claim 1, said method including getHeadersAsXMMAP and getNewHeadersASXMMAP method calls for fetching headers from messages in a mailbox, with the name of the mailbox as input parameter.

5. A method as claimed in claim 1, said method including a getMessagesAsXMMAP method call for retrieving mail message content as information describing the content, with the name of the mailbox and the box number as input parameters.

6. A method as claimed in claim 1, said method including a getAttachmentAsXMMAP method call for retrieving the content of an attachment, with the name of the mailbox, the number of the mailbox and the attachment number as input parameters.

7. A method as claimed in claim 1, said including a delete method call for marking a message as deleted or deleting the message permanently from the email server, with the name and number of the mailbox as input parameters.

8. A method as claimed in claim 1, said method including a sendMail method call for sending mail from the mobile client, with the name and address of the sender, the name and address of the receiver, the subject of the email, the body content of the email as input parameters.

9. A method as claimed in claim 8, said method including to include an attachment content type description, a filename of an attachment, a size of an attachment and a content of an attachment as additional input parameters.

10. A method as claimed in claim 1, said method including a logout method call realized as a message with empty body.

11. A method as claimed in any of the claims 1-10, including compressing the messages.

12. A method as claimed in claim 1 or 5, including to keep back attachments to email messages on the email server until the user operating the mobile terminal send a request for them.

13. A method as claimed in any of the claims 1-10, including caching all messages and attachments in said email services server.

14. A method as claimed in claim 1, including removing unnecessary email headers.

15. A method as claimed in any of the claims 1-10, including encrypting all messages using SSL/TLS.

16. An email communication protocol format for messages to be transferred to and from an mobile terminal, characterized in
a <Headers> element containing a limited set of information relevant for a user of the mobile terminal,
a <Flags> element mapped directly from the IMAP specification,
a <Body> element of "text/plain"-MIME type without alternative representations.

17. A protocol format as claimed in claim 16, said protocol including a <BoxNumber> element representing a message's number in a mailbox in the email server.

18. A protocol format as claimed in claim 16, said protocol including an <Attachment> element with
a <Filename> sub-element containing the name of an attached file,
a <Size> sub-element containing the size of the file, and
a <Content> sub-element that is empty.

19. A system for mobile email communication, characterized in
an email server with POP/IMAP/SMTP interfaces,
a mobile terminal with an email client with a SOAP interface,
an email services server with POP/IMAP/SMTP interfaces, as well as a first SOAP interface,
said email services server being arranged to interpret XML protocol format message requests from the mobile terminal received on said first SOAP interface, convert said message requests into POP/IMAP/SMTP format messages, send the converted messages to the email server, convert the result into SOAP messages and send the SOAP messages too the mobile client.

20. A system as claimed in claim 1,
wherein said email services server includes additional SOAP interfaces for communication with other Web servers.

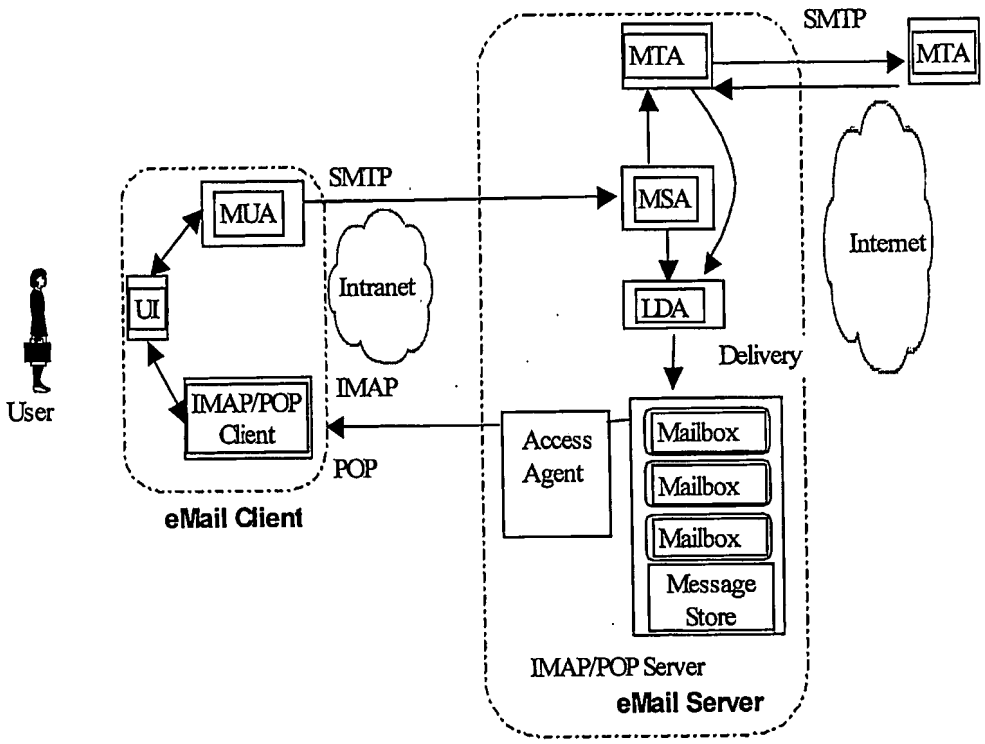

SMTP
MTA
MTA
MUA
SMTP
MSA
Internet
Intranet
UI
LDA
Delivery
User
IMAP/POP
Client
IMAP
POP
Access
Agent
Mailbox
Mailbox
Mailbox
Message
Store
eMail Client
IMAP/POP Server
eMail Server

Figure 1

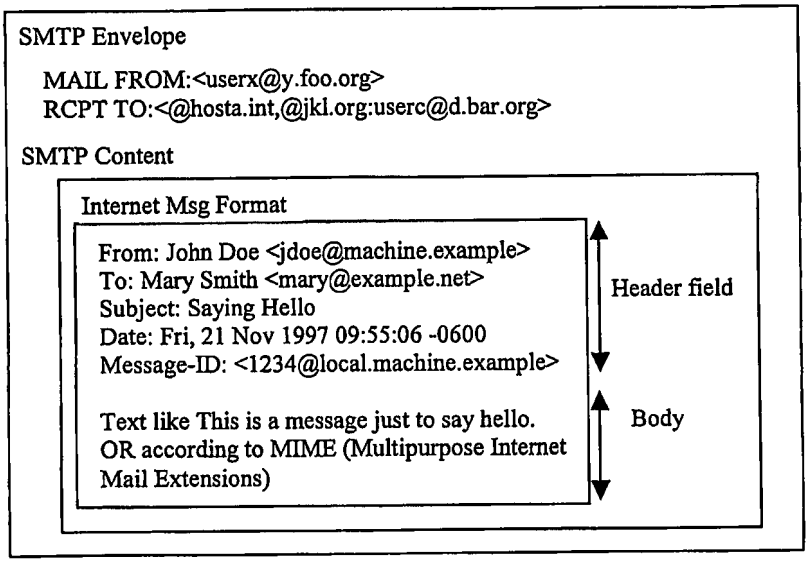

SMTP Envelope
MAIL FROM:<userx@y.foo.org>
RCPT TO:<@hosta.int,@jkl.org:userc@d.bar.org>
SMTP Content
Internet Msg Format
From: John Doe <jdoe@machine.example>
To: Mary Smith <mary@example.net>
Subject: Saying Hello
Date: Fri, 21 Nov 1997 09:55:06 -0600
Message-ID: <1234@local.machine.example>
Text like This is a message just to say hello.
OR according to MIME (Multipurpose Internet Mail Extensions)
Header field
Body

Figure 2

SMTP - mailsending proceedure (SMTP Envelope)

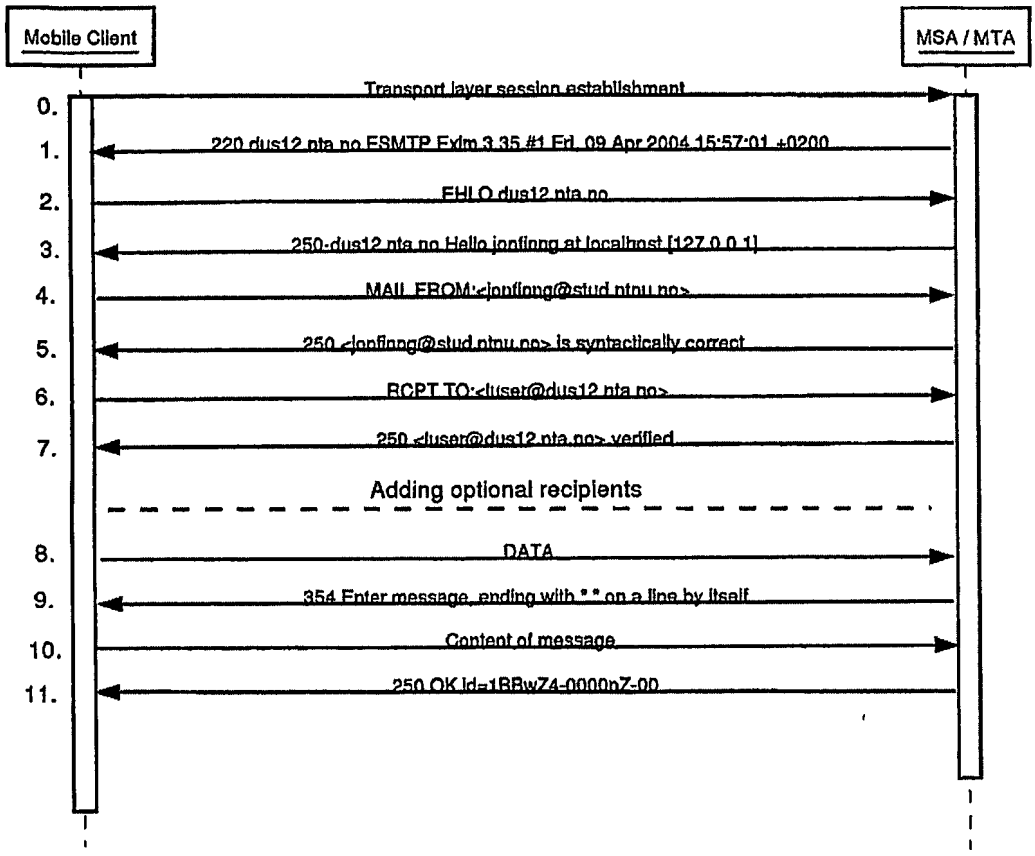


Figure 3

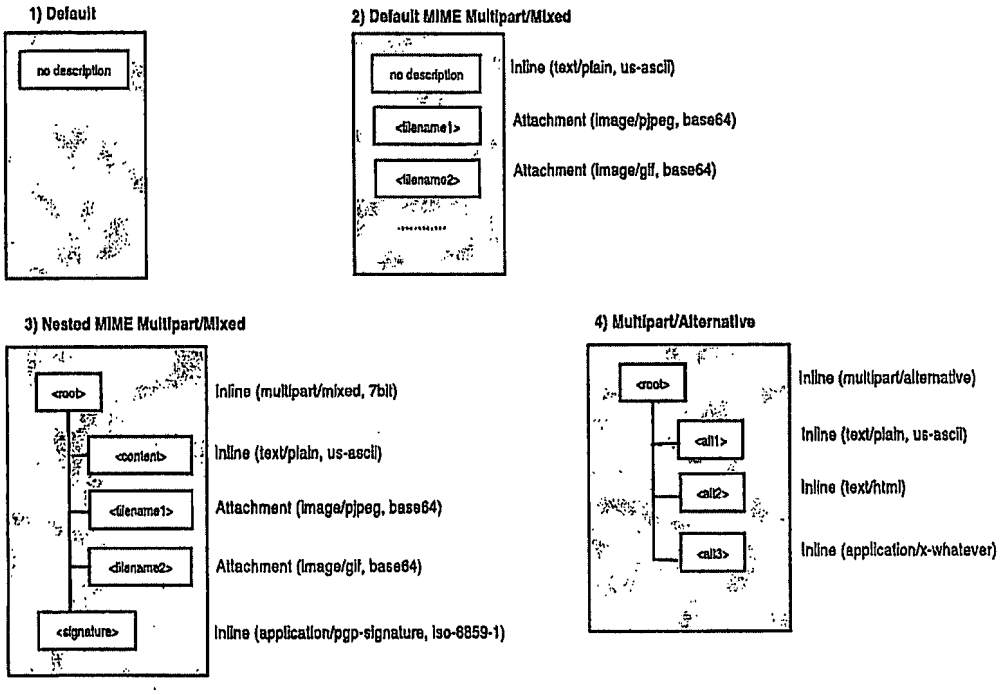

E-mail part organization
1) Default
no description
2) Default MIME Multipart/Mixed
no description
<filename1>
<filename2>
Inline (text/plain, us-ascii)
Attachment (image/pjpeg, base64)
Attachment (image/gif, base64)
3) Nested MIME Multipart/Mixed
<root>
<content>
<filename1>
<filename2>
<signature>
Inline (multipart/mixed, 7bit)
Inline (text/plain, us-ascii)
Attachment (image/pjpeg, base64)
Attachment (image/gif, base64)
Inline (application/pgp-signature, iso-8859-1)
4) Multipart/Alternative
<root>
<alt1>
<alt2>
<alt3>
Inline (multipart/alternative)
Inline (text/plain, us-ascii)
Inline (text/html)
Inline (application/x-whatever)

Figure 4

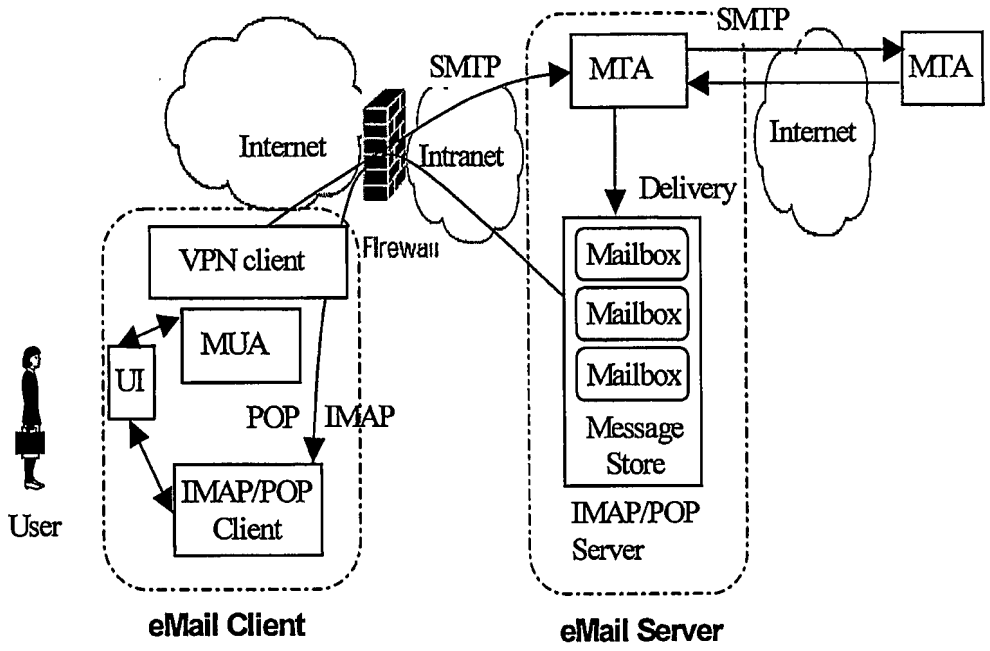

SMTP
MTA
MTA
SMTP
Internet
Intranet
Internet
Delivery
VPN client
Firewall
Mailbox
Mailbox
Mailbox
MUA
UI
POP
IMAP
Message
Store
IMAP/POP
Client
IMAP/POP
Server
User
eMail Client
eMail Server

Figure 5

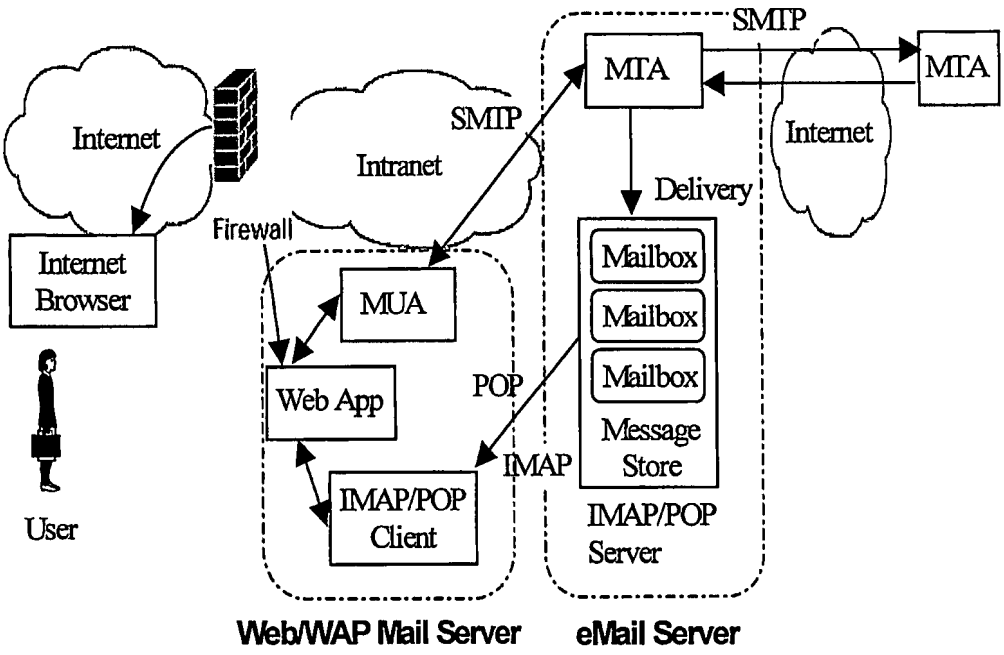

SMTP
MTA
MTA
Internet
Internet
SMTP
Intranet
Delivery
Firewall
Internet
Browser
Mailbox
Mailbox
Mailbox
MUA
Web App
POP
Message
Store
IMAP
IMAP/POP
Client
IMAP/POP
Server
User
Web/WAP Mail Server
eMail Server

Figure 6

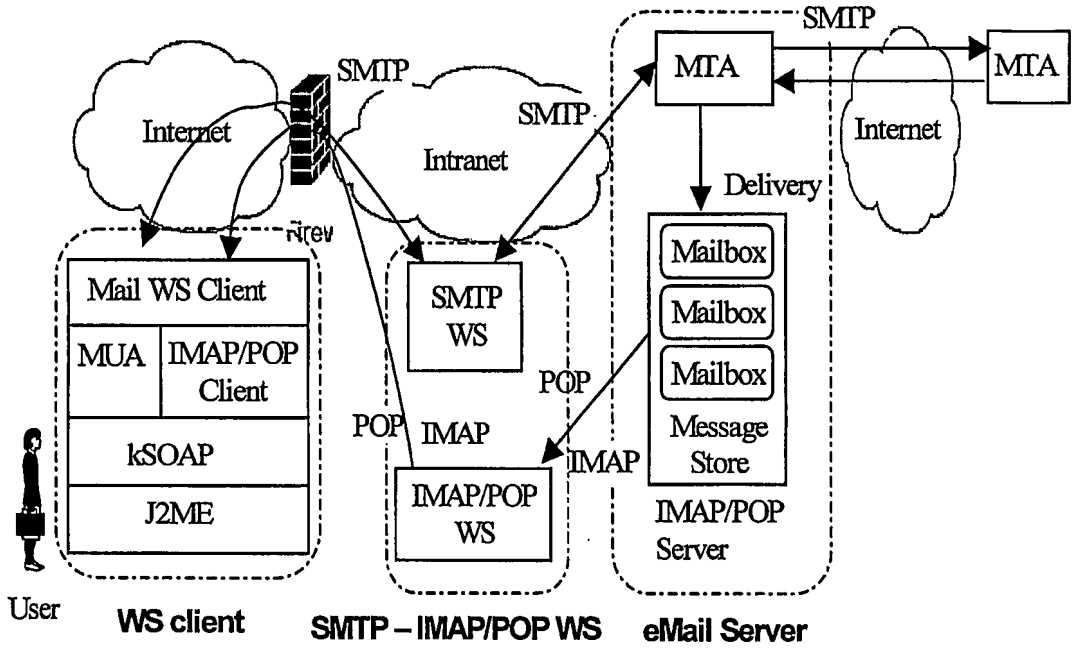

SMTP
MTA
MTA
Internet
SMTP
SMTP
Intranet
Internet
Delivery
Mailbox
Mailbox
Mailbox
Message Store
IMAP/POP Server
Mail WS Client
MUA
IMAP/POP Client
kSOAP
J2ME
SMTP WS
POP
POP
IMAP
IMAP
IMAP/POP WS
User
WS client
SMTP – IMAP/POP WS
eMail Server

Figure 7

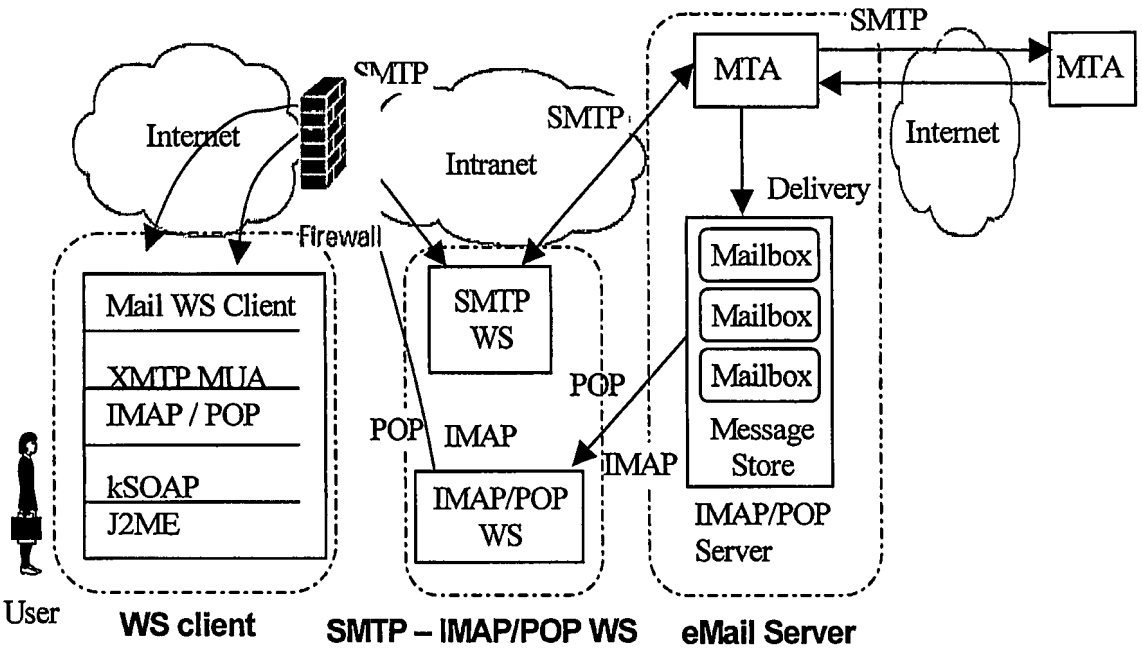

SMTP
MTA
MTA
Internet
SMTP
Intranet
SMTP
Internet
Delivery
Firewall
Mail WS Client
XMTP MUA
IMAP / POP
kSOAP
J2ME
SMTP
WS
POP
POP
IMAP
IMAP
IMAP/POP
WS
Mailbox
Mailbox
Mailbox
Message
Store
IMAP/POP
Server
User
WS client
SMTP – IMAP/POP WS
eMail Server

Figure 8

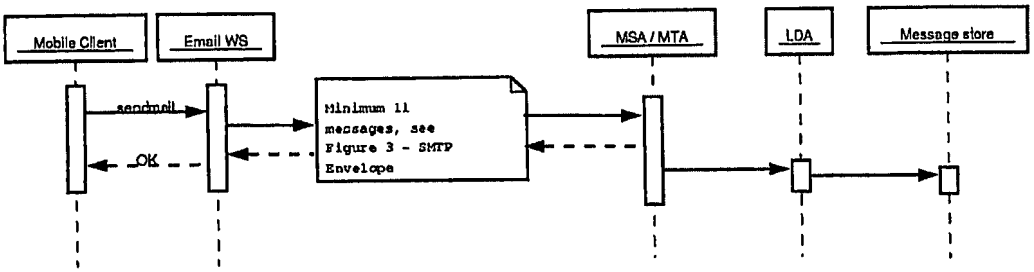

Mobile Client
Email WS
MSA / MTA
LDA
Message store
sendmail
OK
Minimum 11
messages, see
Figure 3 - SMTP
Envelope

Figure 9

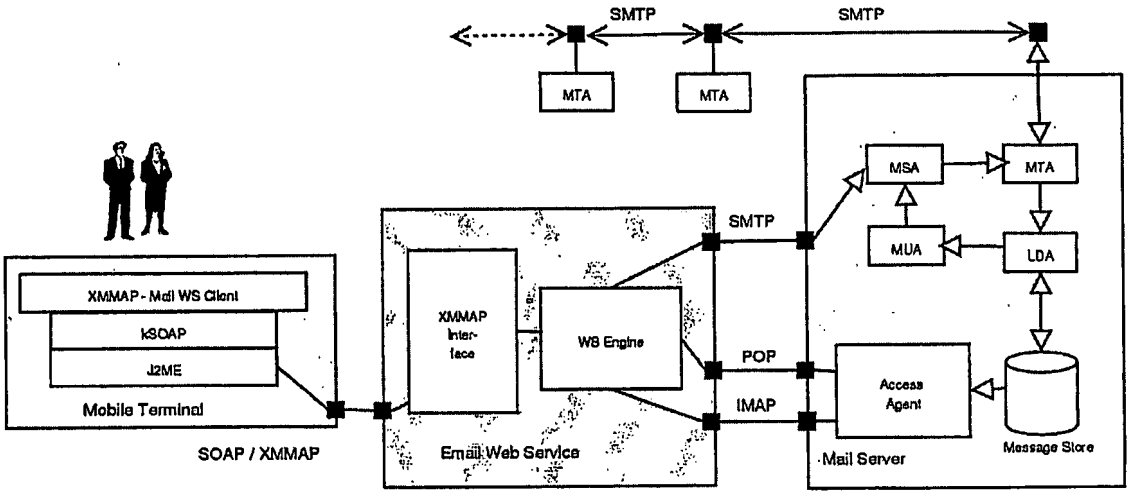

SMTP
SMTP
MTA
MTA
MSA
MTA
MUA
LDA
SMTP
XMMAP - Mail WS Client
kSOAP
J2ME
Mobile Terminal
XMMAP Inter-face
WS Engine
POP
IMAP
Access Agent
Message Store
SOAP / XMMAP
Email Web Service
Mail Server

Figure 10

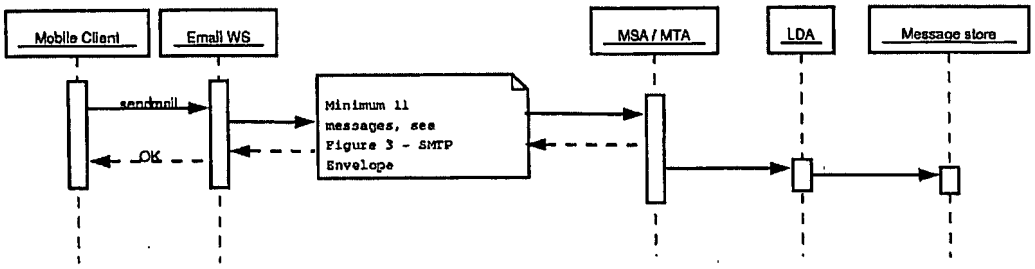

Mobile Client
Email WS
MSA / MTA
LDA
Message store
sendmail
OK
Minimum 11
messages, see
Figure 3 - SMTP
Envelope

Figure 11

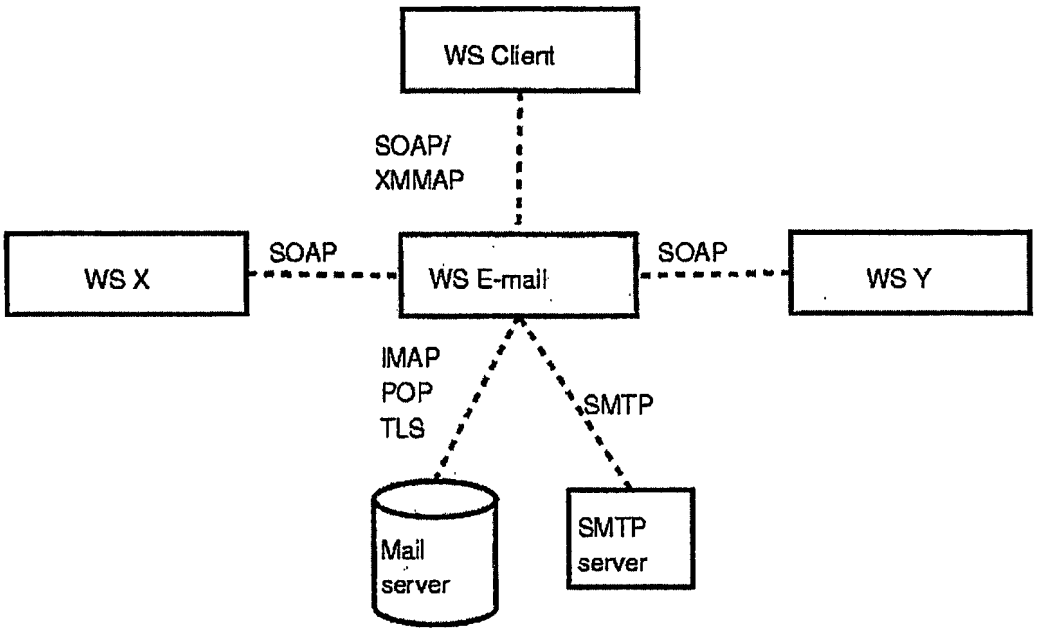

WS Client
SOAP/
XMMAP
WS X
SOAP
WS E-mail
SOAP
WS Y
IMAP
POP
TLS
SMTP
Mail
server
SMTP
server

Figure 12

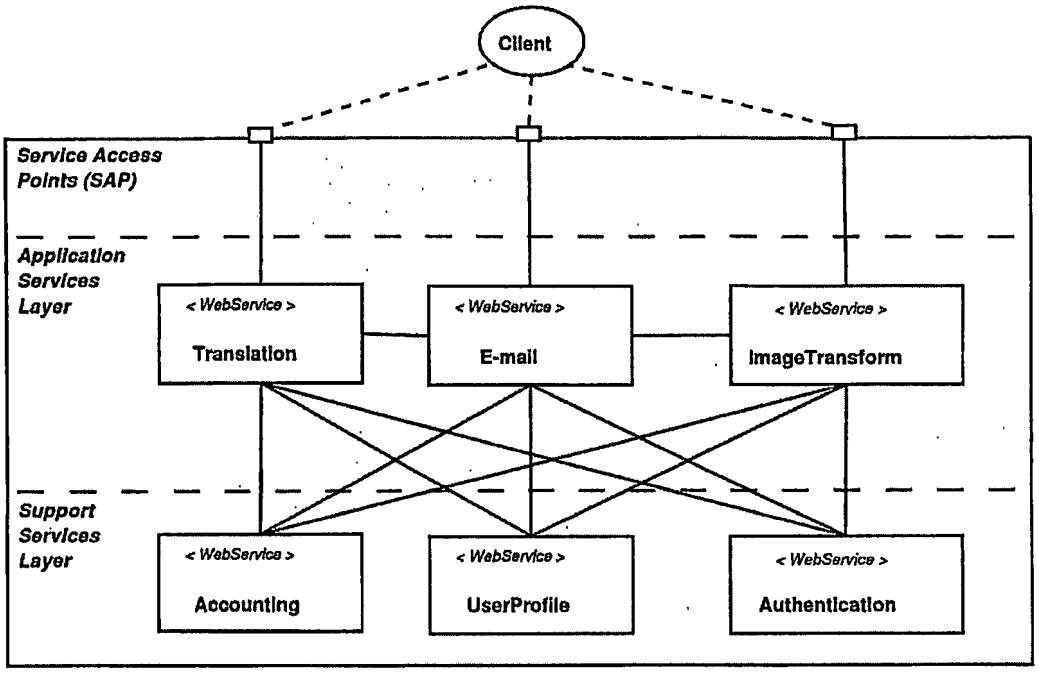

Client
Service Access Points (SAP)
Application Services Layer
< WebService >
Translation
< WebService >
E-mail
< WebService >
ImageTransform
Support Services Layer
< WebService >
Accounting
< WebService >
UserProfile
< WebService >
Authentication

Figure 13

INTERNATIONAL SEARCH REPORT

International Application No
PCT/NO2005/000175

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L12/58 H04L29/08

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, PAJ, WPI Data, INSPEC, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2003/158902 A1 (VOLACH DOTAN) 21 August 2003 (2003-08-21) paragraph '0031! - paragraph '0035! paragraph '0048!	1-20
Y	BORDEN J: "XML MIME Transformation Protocol (XMTP)"'Online! 6 April 2004 (2004-04-06), pages 1-4, XP002342219 Retrieved from the Internet: URL:http://web.archive.org/web/20040406050718/http://www.openhealth.org/documents/xmtp.htm> 'retrieved on 2005-08-18! cited in the application the whole document	1-20

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
25 August 2005	07/09/2005

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Ströbeck, A.

INTERNATIONAL SEARCH REPORT

International Application No
PCT/NO2005/000175

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,X	MOE J-F ET AL: "Adapting Email Functionality for Mobile Terminals" PROCEEDINGS OF IFIP INTERNATIONAL CONFERENCE ON INTELLIGENCE IN COMMUNICATION SYSTEMS, INTELLCOMM 2004, 23 November 2004 (2004-11-23), pages 199-206, XP008051312 Berlin, Germany page 201, line 15 - page 205, line 19	1-20
A	US 2003/028606 A1 (KOOPMANS CHRIS ET AL) 6 February 2003 (2003-02-06) figure 3A	11
A	EP 1 284 570 A (RESEARCH IN MOTION LIMITED) 19 February 2003 (2003-02-19) paragraph '0085! paragraph '0115! - paragraph '0121!	11,15
A	US 2002/116465 A1 (KIM SOON-JIN ET AL) 22 August 2002 (2002-08-22) paragraph '0023! - paragraph '0025!	12,13
A	CRISPIN M: "Internet Message Access Protocol - version 4rev1, RFC 2060" IETF, REQUEST FOR COMMENTS, December 1996 (1996-12), XP002122132 page 5, line 41 - page 17, line 42	1-20

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/NO2005/000175

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2003158902 A1	21-08-2003	EP 1451703 A1	01-09-2004
		WO 03038636 A1	08-05-2003
US 2003028606 A1	06-02-2003	EP 1393517 A2	03-03-2004
		WO 02093867 A2	21-11-2002
EP 1284570 A	19-02-2003	US 2002049818 A1	25-04-2002
		CA 2389978 A1	13-02-2003
		EP 1284570 A2	19-02-2003
US 2002116465 A1	22-08-2002	KR 2002067803 A	24-08-2002
		CN 1372207 A	02-10-2002